Maps in R

Diego López Tamayo [*]

Contents

```
pacman::p_load(tidyverse,
               ggplot2,
               sf,
               sp,
               tmap)
```

Why should you use maps?

Maps are used in a variety of fields to express data in an appealing and interpretive way. Data can be expressed into simplified patterns, and this data interpretation is generally lost if the data is only seen through a spread sheet. Maps can add vital context by incorporating many variables into an easy to read and applicable context. Maps are also very important in the information world because they can quickly allow the public to gain better insight so that they can stay informed. It's critical to have maps be effective, which means creating maps that can be easily understood by a given audience.

Basic elements of a map that should be considered are polygon, points, lines, and text.

- Polygons, on a map, are closed shapes such as country borders.
- Lines are considered to be linear shapes that are not filled with any aspect, such as highways, streams, or roads.
- Finally, points are used to specify specific positions, such as city or landmark locations.

Using R to create maps brings these benefits to mapping. Elements of a map can be added or removed with ease — R code can be tweaked to make major enhancements with a stroke of a key. It is also easy to reproduce the same maps for different data sets.

The package ggplot2 implements the grammar of graphics in R, as a way to create code that make sense to the user: The grammar of graphics is a term used to breaks up graphs into semantic components, such as geometries and layers. Practically speaking, it allows (and forces!) the user to focus on graph elements at a higher level of abstraction, and how the data must be structured to achieve the expected outcome. Recently, the package ggplot2 has allowed the use of simple features from the package sf as layers in a graph1. The combination of ggplot2 and sf therefore enables to programmatically create maps, using the grammar of graphics.

[*]El Colegio de México, diego.lopez@colmex.mx

tmap function

With the tmap package, thematic maps can be generated with great flexibility. The syntax for creating plots is similar to that of ggplot2, but tailored to maps.

First steps

A good place to start is to create a map of the world.

The object World is a spatial object of class sf from the sf package; it is a data.frame with a special column that contains a geometry for each row, in this case polygons. In order to plot it in tmap, you first need to specify it with tm_shape. Layers can be added with the + operator, in this case tm_polygons. There are many layer functions in tmap, which can easily be found in the documentation by their tm_ prefix.

```
data("World")
class(World)
```

```
## [1] "sf"         "data.frame"
```

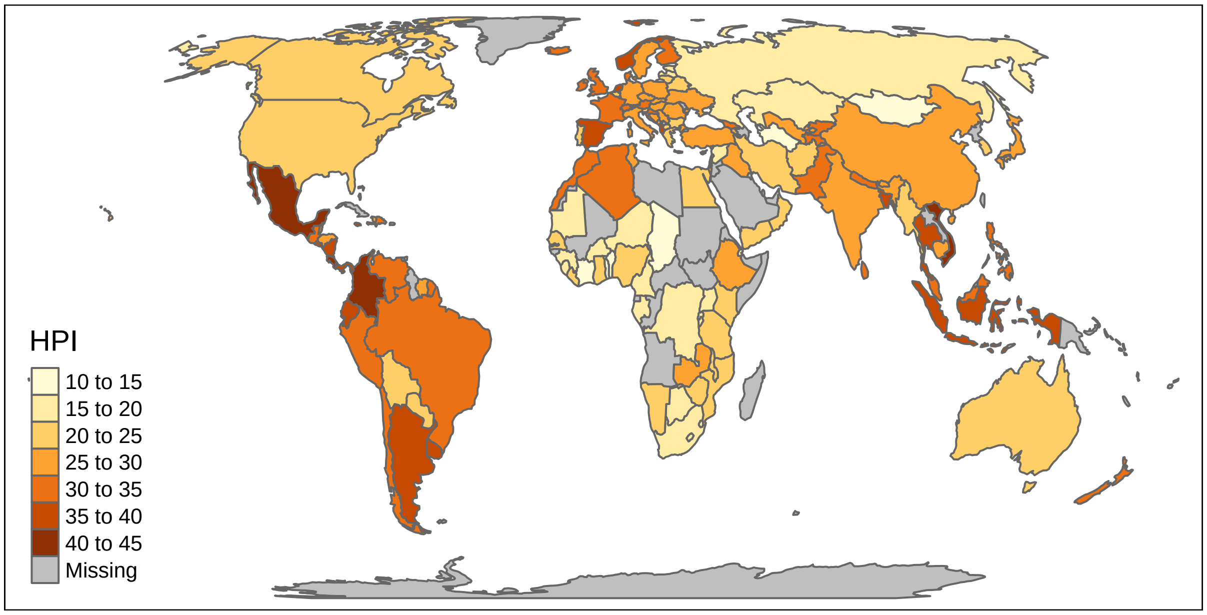
The "World"-DataFrame includes informations on the population, the economy and so on of all countries in the world.

```
World %>% as.data.frame() %>% head()
```

```
##   iso_a3                 name           sovereignt     continent
## 1    AFG          Afghanistan          Afghanistan          Asia
## 2    AGO               Angola               Angola        Africa
## 3    ALB              Albania              Albania        Europe
## 4    ARE United Arab Emirates United Arab Emirates          Asia
## 5    ARG            Argentina            Argentina South America
## 6    ARM              Armenia              Armenia          Asia
##                area  pop_est pop_est_dens                    economy
## 1  652860.00 [km^2] 28400000     43.50090 7. Least developed region
## 2 1246700.00 [km^2] 12799293     10.26654 7. Least developed region
## 3   27400.00 [km^2]  3639453    132.82675      6. Developing region
## 4   71252.17 [km^2]  4798491     67.34519      6. Developing region
## 5 2736690.00 [km^2] 40913584     14.95003   5. Emerging region: G20
## 6   28470.00 [km^2]  2967004    104.21510      6. Developing region
##                income_grp gdp_cap_est life_exp well_being footprint inequality
## 1           5. Low income    784.1549   59.668        3.8      0.79  0.4265574
## 2  3. Upper middle income   8617.6635       NA         NA        NA         NA
## 3  4. Lower middle income   5992.6588   77.347        5.5      2.21  0.1651337
## 4 2. High income: nonOECD  38407.9078       NA         NA        NA         NA
## 5  3. Upper middle income  14027.1261   75.927        6.5      3.14  0.1642383
## 6  4. Lower middle income   6326.2469   74.446        4.3      2.23  0.2166481
##        HPI                       geometry
## 1 20.22535 MULTIPOLYGON (((5310471 451...
## 2       NA MULTIPOLYGON (((1531585 -77...
## 3 36.76687 MULTIPOLYGON (((1729835 521...
## 4       NA MULTIPOLYGON (((4675864 313...
## 5 35.19024 MULTIPOLYGON (((-5017766 -6...
## 6 25.66642 MULTIPOLYGON (((3677241 513...
```

After installing tmap, the following lines of code should create the map shown below:

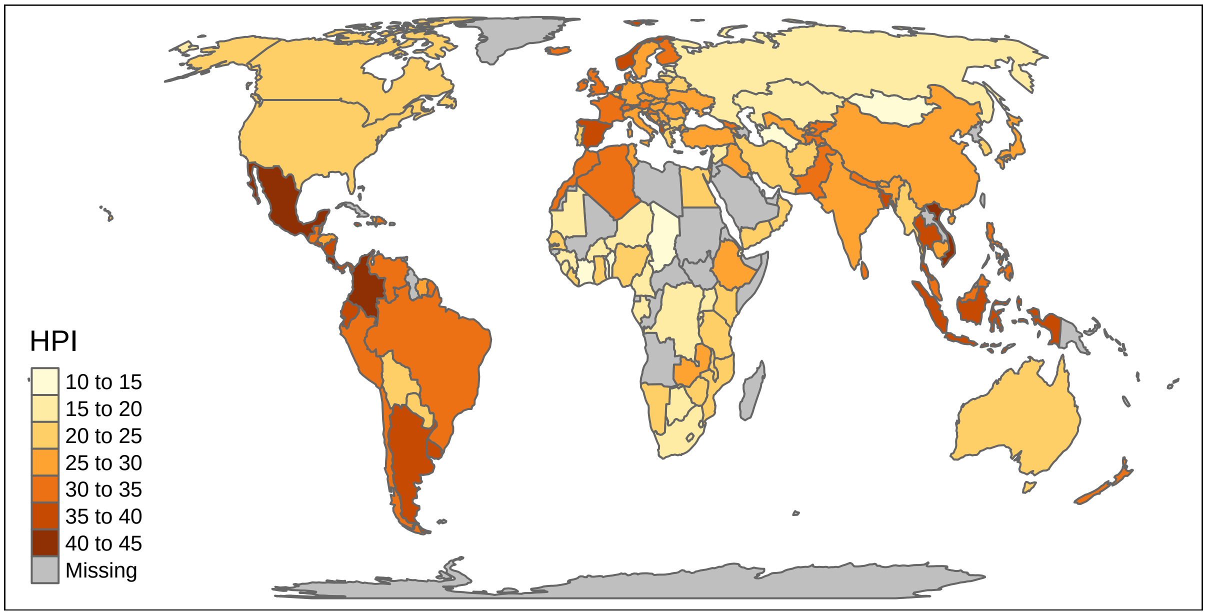
```
tmap_mode("plot")
tm_shape(World) +
    tm_polygons("HPI")
```



Interactive maps

Each map can be plotted as a static image or viewed interactively using “plot” and “view” modes, respectively. The mode can be set with the function tmap_mode, and toggling between the modes can be done with the ‘switch’ ttm() (which stands for toggle thematic map.

```
tmap_mode("plot")
tm_shape(World) +
    tm_polygons("HPI")
```



Multiple shapes and layers

A shape is a spatial object (with a class from sf, sp, stars, or raster). Multiple shapes and also multiple layers per shape can be plotted.

```r
data(World, metro)
```

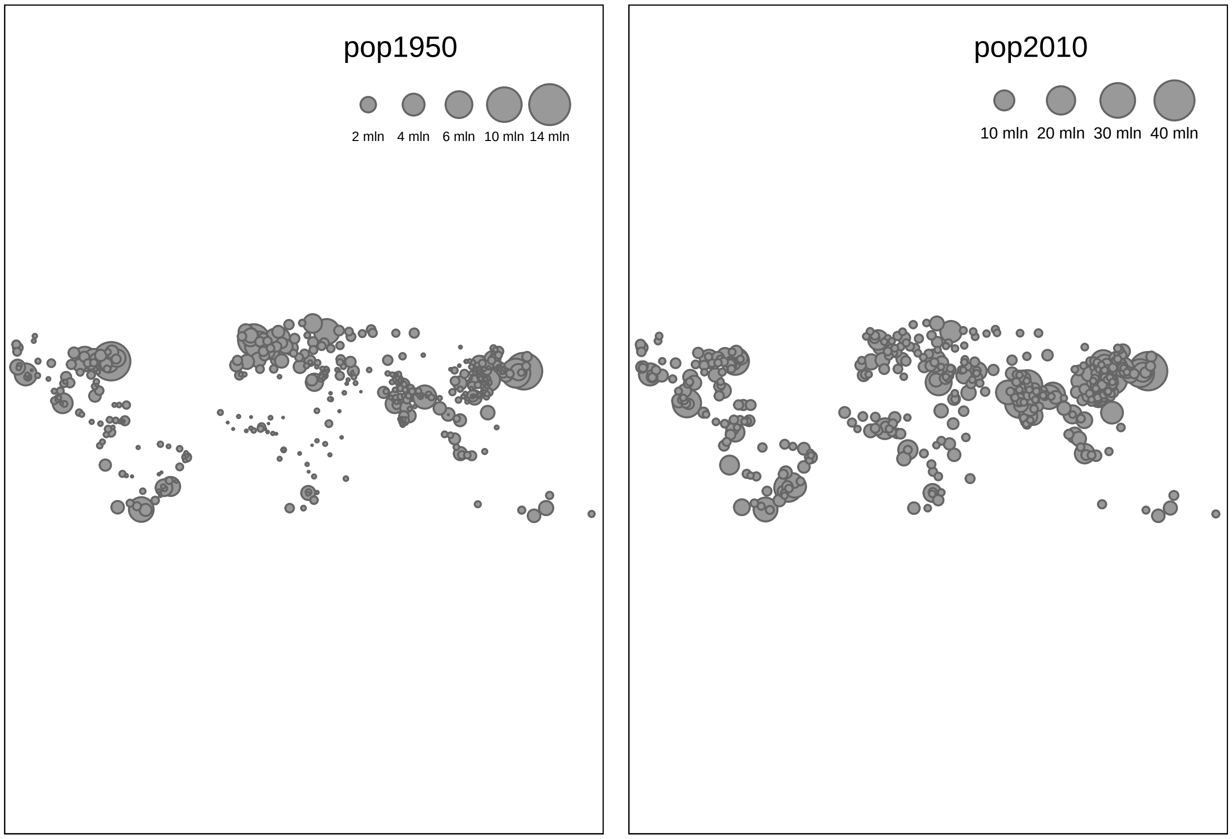
Let's have a look into the metro-Data Frame. It shows the population of the metropolitan regions in the world, for example all metropolitan regions of Switzerland, i.e. Zurich.

```r
metro %>% as.data.frame() %>% dplyr::filter(iso_a3=="CHE") %>% head()
```

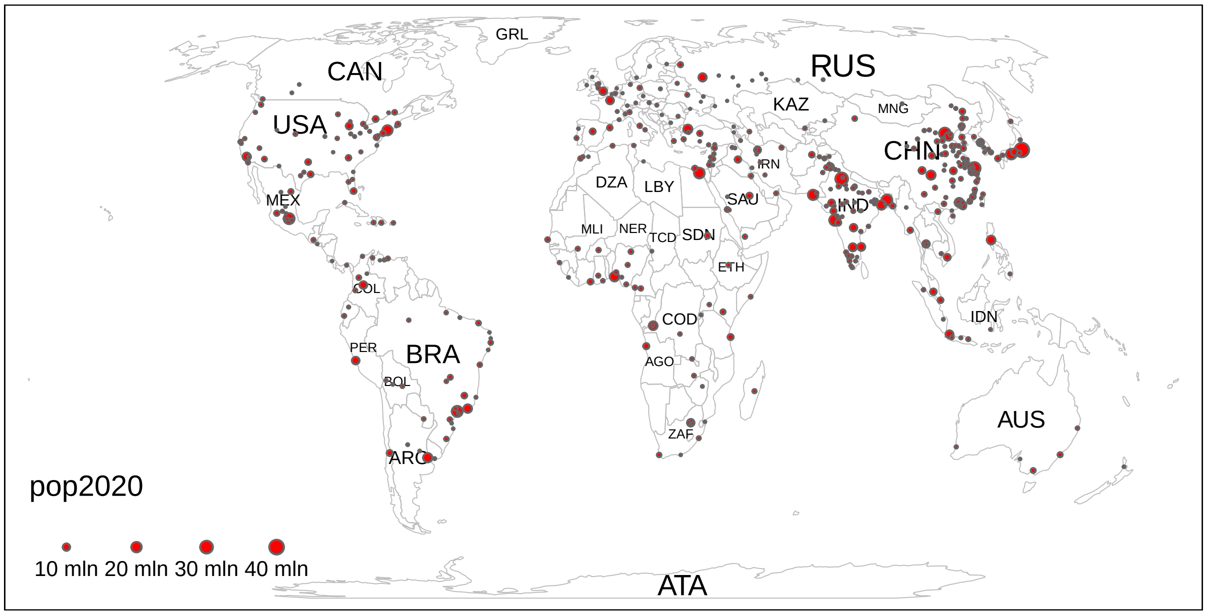
```
##     name       name_long iso_a3 pop1950 pop1960 pop1970 pop1980 pop1990 pop2000
## 1 Zurich Zurich (Zurich)    CHE  493648  535471  710580  706766 1005859 1078135
##   pop2010 pop2020 pop2030                 geometry
## 1 1186491 1323169 1493733 POINT (8.50296 47.35785)
```

Let's look at the whole world population in 1950 and 2010.

```r
tmap_mode("plot")
map1<-tm_shape(metro)+tm_bubbles("pop1950")
map2<-tm_shape(metro)+tm_bubbles("pop2010")
tmap_arrange(map1,map2,nrow=1)
```

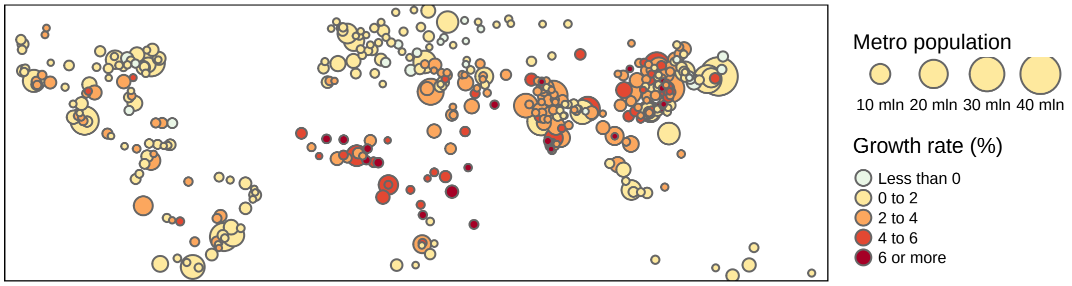


```r
tmap_mode("plot")
## tmap mode set to plotting
#tm_shape(land) +
    #tm_raster("elevation", palette = terrain.colors(4)) +
tm_shape(World) +
    tm_borders("gray", lwd = .5) +
    tm_text("iso_a3", size = "AREA") +
tm_shape(metro) +
    tm_symbols(col = "red", size = "pop2020", scale = .5) +
tm_legend(show = T)
```



A much better picture can be obtained if the information on the population is presented in a somewhat better graphic form and supplemented by a growth rate of the population figures. So, let's add some color and a nicer looking legend.

```
# Add the column growth into metro dataset
metro$growth <- (metro$pop2020 - metro$pop2010) / (metro$pop2010 * 10) * 100
tm_shape(metro) +
tm_bubbles("pop2010", col = "growth",
 breaks=c(-Inf, seq(0, 6, by=2), Inf),
               palette="-RdYlBu", contrast=1,
               title.size="Metro population",
               title.col="Growth rate (%)", id="name")+
  tm_layout(legend.outside = T)
```



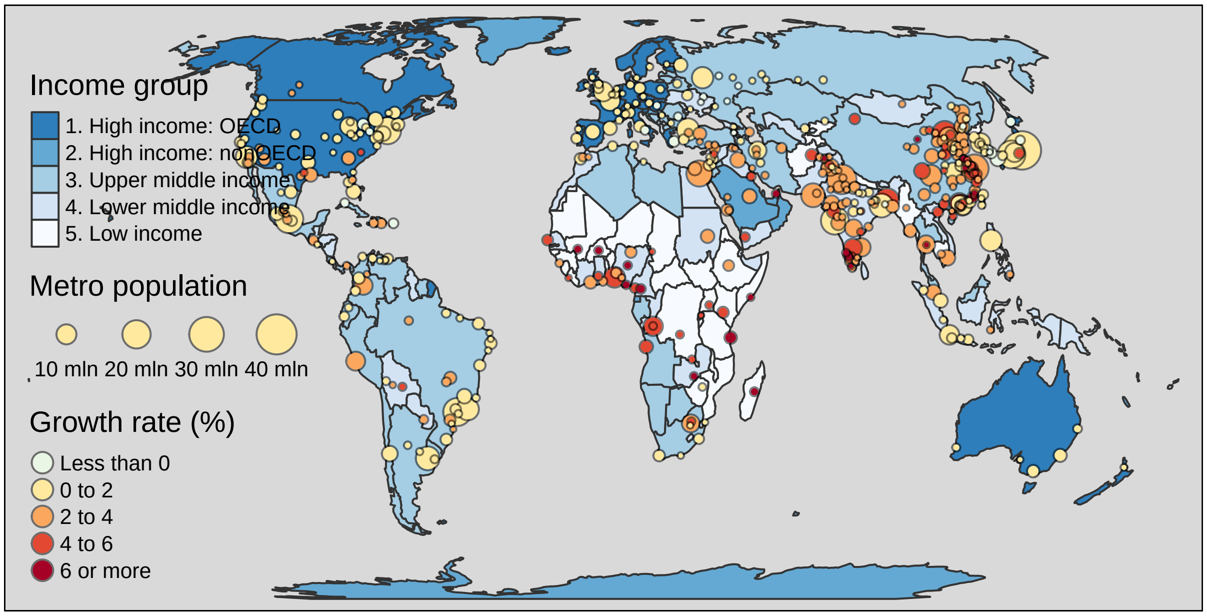
Now we can simply put the two maps (the happiness index world map an the population growth bubbles) on top of each other and get a simple yet informative map.

```
data(World, metro)
metro$growth <- (metro$pop2020 - metro$pop2010) / (metro$pop2010 * 10) * 100

## Define global style and format for map
tmap_style("gray")
tmap_format("World_wide")
```

```
## $inner.margins
## [1] 0.000 0.200 0.025 0.010
```

```
##
## $legend.position
## [1] "left"   "bottom"
##
## $attr.position
## [1] "right"  "bottom"
##
## $scale
## [1] 0.8
```

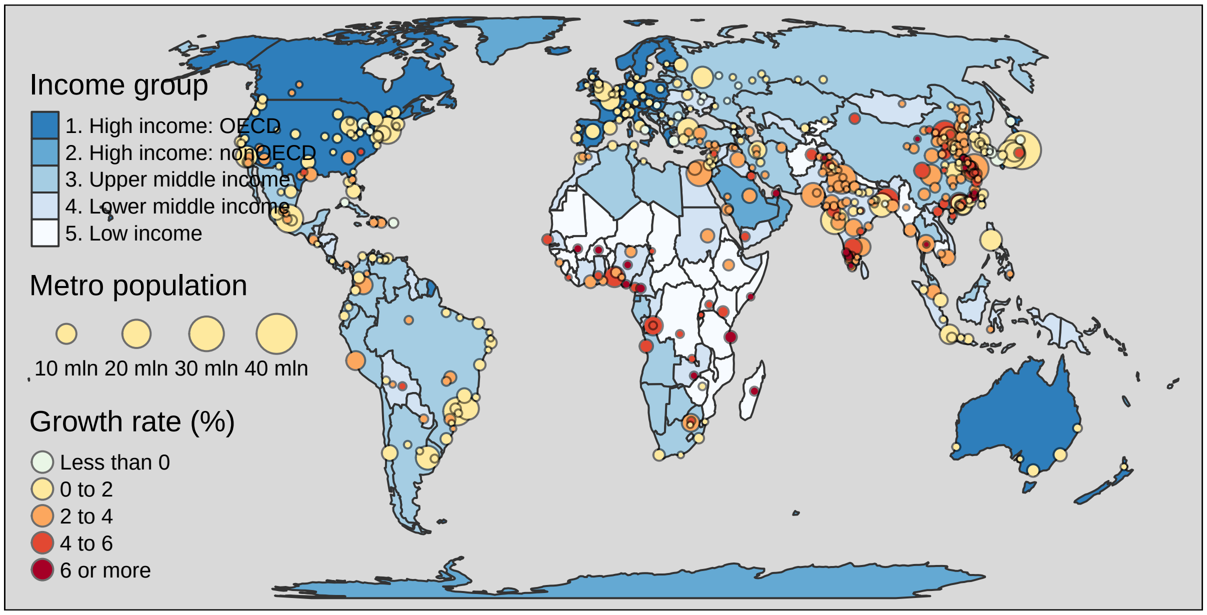
```
mapWorld <- tm_shape(World) +
    tm_polygons("income_grp", palette="-Blues", contrast=.7, id="name", title="Income group") +
    tm_shape(metro) +
    tm_bubbles("pop2010", col = "growth",
               border.col = "black", border.alpha = .5,
               style="fixed", breaks=c(-Inf, seq(0, 6, by=2), Inf),
               palette="-RdYlBu", contrast=1,
               title.size="Metro population",
               title.col="Growth rate (%)", id="name")

mapWorld
```



So this is our first static map. You can't see details, but you can already see a lot of things from an overview point of view. But if you want to look at details of individual regions or metropolises, you need something more, namely an interactive version, which allows you to zoom into your point of interest.

This is very easy with the tmap package. The only thing to do is to change from the plot mode to the view mode. Afterwards we easyly plot the existing map a second time. Nad abracadabra you have an interactive leaflet map, where you can zoom in and some additional informations in a tooltip.

```
# set mode to view
# Play with the layers
tmap_mode("plot")
mapWorld
```


Income group
1. High income: OECD
2. High income: nonOECD
3. Upper middle income
4. Lower middle income
5. Low income
Metro population
10 mln 20 mln 30 mln 40 mln
Growth rate (%)
Less than 0
0 to 2
2 to 4
4 to 6
6 or more